Identificazione degli scribi nella Bibbia di Avila

Davide Pucci Matricola: 6283923

31 gennaio 2020

Indice

1 Introduzione

In questo esperimento si è cercato di riprodurre i risultati ottenuti in *De Stefano et al. 2018* utilizzando l'implementazione fornita da **scikit-learn** di Decision Tree per identificare lo scriba responsabile della copiatura di un paragrafo nella Bibbia di Avila.

1.1 Il Dataset

Abbiamo utilizzato l'*Avila Dataset* disponibile online per allenare e poi testare il DTree. Esso si trova con dati già normalizzati secondo la *Z-Normalization* e comprende 20867 campioni già suddivisi tra training e test:

- **Training Set**: 10430 campioni.
- **Test Set**: 10437 campioni.

Ogni campione è costituito da 10 *features* ottenute dall'analisi grafica di circa 4 righe della Bibbia (un paragrafo), più la *classe* di appartenenza.
Ciascuna feature offre una informazione su come è stato scritto il paragrafo in analisi, come ad esempio il peso del tratto o la gestione degli spazi bianchi. Le classi di appartenenza sono 12, non sono equamente distribuite e sono indicate da una lettera: A, B, C, D, E, F, G, H, I, W, X, Y.

1.2 Il Codice

Il codice, scritto in Python, è piuttosto semplice, fa ampio uso di **scikit-learn**, il suo lavoro può essere sintetizzato in 5 passi:

1. Apre il file `avila-tr.txt` contenente il training set, lo legge e carica in memoria due array:
   - X: contiene tutte le features dei campioni di training.
   - Y: contiene la classe di appartenenza di ciascun campione.
2. Effettua 10-fold cross-validation per determinare il miglior valore per l'iperparametro N_l (minimo numero di campioni necessari per creare una foglia) utilizzando il metodo `sklearn.model_selection.cross_val_score()` e poi produce `Nl.csv` in cui vi sono i risultati ottenuti.
3. Crea il DTree utilizzando il parametro ottenuto precedentemente e l'*Entropia* come criterio di classificazione, poi vi aggiunge X e Y.
4. Apre il file `avila-ts.txt` contenente il test set e lo utilizza per testare il DTree, mantenendo i risultati ottenuti in 3 dizionari.
5. Produce quindi in output `results.csv` contenente il grado di precisione e di richiamo per ciascuna classe.

2 Risultati Ottenuti

Nell'articolo *De Stefano et al. 2018* si utilizza **Weka**, in particolare l'algoritmo *J48* per la creazione del DTree, mentre **scikit-learn**, utilizzato in questo caso, offre una versione ottimizzata dell'algoritmo *CART* per la creazione del DTree. Per far lavorare l'algoritmo *CART* in maniera più simile a *J48* si è fatto uso dell'*Entropia* come criterio per ottenere il maggior *Information-Gaining*, invece di *Gini* che è il criterio di default. I risultati ottenuti sono i seguenti.

2.1 Iperparametro N_l

Il minimo numero di campioni necessari alla creazione di una foglia costituisce quello che nell'articolo di riferimento viene indicato come l'iperparametro N_l. Nell'esperimento originale, si cercava N_l in un range di valori tra 1 e 5, anche in relazione al fattore di confidenza C_f che influenza il *post-pruning*.

Qui non abbiamo preso in considerazione il *post-pruning* e ci siamo concentrati solo su N_l, effettuando 10-fold cross-validation come nell'articolo originale. Per **scikit-learn** l'iperparametro N_l corrisponde a *min_samples_leaf*, che è stato variato da 1 a 5, registrando il grado di accuratezza ottenuto in ciascun caso. Nella tabella 1 vediamo i risultati del test.

N	1	2	3	4	5
Score	0.987440	0.983988	0.980824	0.979769	0.974880

Tabella 1: Score ottenuti con la 10-fold cross-validation al variare del parametro N_l (*min_samples_leaf*).

Come vediamo, nel caso di $N_l = 1$, abbiamo ottenuto i risultati migliori. Nell'articolo di riferimento si è cercato di massimizzare lo score per la coppia (C_f, N_l), ottenendo come risultato migliore $(0.25, 2)$; questo, oltre che massimizzare su due parametri, tiene conto anche dell'efficacia del *post-pruning*, per cui il fatto di ottenere due risultati lievemente diversi è possibile che sia dovuto a questo fattore, oltre al fatto di utilizzare due algoritmi differenti.

2.2 Precisione e Richiamo

Una volta scelto N_l, si è creato il Decision Tree usando *Entropia* come criterio per l'information-gaining. Con la classe `sklearn.tree.DecisionTreeClassifier` si è creato il classificatore e vi si è aggiunto il training set.

Nello specifico si è cercato di replicare i risultati ottenuti nella Tabella 7 dell'articolo *De Stefano et al. 2018*, in Tabella 2 sono contenuti i risultati ottenuti.

Writer	prec.	rec.
A	98.0	99.6
B	66.7	80.0
C	95.2	97.1
D	97.2	97.5
E	96.7	97.4
F	98.9	98.9
G	98.4	97.5
H	98.0	95
I	100	98.8
W	100	97.8
X	96.9	88.9
Y	100	98.9

Writer	Precision	Recall
A	99.416	99.323
B	80.0	80.0
C	97.142	99.029
D	97.435	96.883
E	96.654	97.625
F	99.089	99.796
G	99.297	94.854
H	97.884	97.884
I	100.0	99.879
W	100.0	100.0
X	94.876	95.785
Y	98.473	96.629

Tabella 2: A **sinistra** i risultati dell'articolo di riferimento, a **destra** quelli ottenuti sperimentalmente.

Come si nota i risultati sono analoghi in ogni classe, compreso B, dove a causa dei pochi campioni presenti, il DTree non è stato molto accurato.

3 Utilizzo del codice

Basta eseguire con Python 3 lo script `code.py`, assicurandosi che nella directory dove viene eseguito vi siano i due file del dataset (`avila-tr.txt` e `avila-ts.txt`). Nella stessa directory, dopo aver eseguito il codice, vengono prodotti i due `.csv` che contengono i risultati ottenuti.